Proceedings of the Third International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2016) Sofia, Bulgaria

Jesus Carretero, Javier Garcia Blas, Svetozar Margenov
(Editors)

October, 6-7, 2016

Ristov, S., Prodan, R., Gusev, M., Petcu, D. & Barbosa, J. (2016). Elastic Cloud Services Compliance with Gustafson's and Amdahl's Laws. En *Proceedings of the Third International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2016) Sofia, Bulgaria* (pp. 1-9). Madrid: Universidad Carlos III de Madrid. Computer Architecture, Communications, and Systems Group (ARCOS).

NESUS cost IC1305
Network for Sustainable Ultrascale Computing
Book paper template • September 2014 • Vol. I, No. 1

Elastic Cloud Services Compliance with Gustafson's and Amdahl's Laws

Sasko Ristov, Radu Prodan

University of Innsbruck, Austria
sasko@dps.uibk.ac.at, radu@dps.uibk.ac.at

Marjan Gusev

Ss. Cyril and Methodius University, Skopje, Macedonia
marjan.gushev@finki.ukim.mk

Dana Petcu

West University of Timisoara, Romania
petcu@info.uvt.ro

Jorge Barbosa

University of Porto, Portugal
jbarbosa@fe.up.pt

Abstract

The speedup that can be achieved with parallel and distributed architectures is limited at least by two laws: the Amdahl's and Gustafson's laws. The former limits the speedup to a constant value when a fixed size problem is executed on a multiprocessor, while the latter limits the speedup up to its linear value for the fixed time problems, which means that it is limited by the number of used processors. However, a superlinear speedup can be achieved (speedup greater than the number of used processors) due to insufficient memory, while, parallel and, especially distributed systems can even slowdown the execution due to the communication overhead, when compared to the sequential one. Since the cloud performance is uncertain and it can be influenced by available memory and networks, in this paper we investigate if it follows the same speedup pattern as the other traditional distributed systems. The focus is to determine how the elastic cloud services behave in the different scaled environments. We define several scaled systems *and we model the corresponding performance indicators. The analysis shows that both laws limit the speedup for a specific range of the input parameters and type of scaling. Even more, the speedup in cloud systems follows the Gustafson's extreme cases, i.e. insufficient memory and communication bound domains.*

Keywords Load, Distributed systems, Performance, Superlinear speedup.

I. Introduction

Cloud computing has introduced a rapid change in the way of designing the architecture of today's services from license-based to as-a-service-based services [1]. The main driver was influenced by its multitenancy, on demand elastic resources and underlined virtualisation technology. Customers do not buy the license to own the software service, but instead they pay only for the period of its usage. In order to satisfy the customers' demands, cloud providers offer various types of resources, usually represented as virtual machine (VM) instances, each with specific computing, memory and storage capacity. The customers expectation is that the performance will follow the price.

Due to its elasticity and the linear pay-as-you-go model, the cloud is preferred platform both for the granular and scalable algorithms, especially if they are low communication-intensive, such as scientific applications [2, 3]. Still, many applications are data-intensive, and provide a high throughput. This is a huge challenge in the cloud because the data

transfer between the cloud compute nodes and storage is a bottleneck [4]. Despite the additional virtualisation layer, the superlinear speedup is also reported, both for granular [5], and scalable application types [6].

However, despite all these benefits, the main challenge for the customers is whether they will get the performance proportionally to the cost. That is, whether the cloud elastic resources comply with the Amdahl's Law [7] for the fixed size problems and with Gustafson's Law [8] for the fixed time problems. In this paper, we model several performance indicators, to determine if both laws hold for the cloud elastic services, each in a specific region. Although one can argue that the web services are scalable and therefore will comply with the Gustafson's law only, our analysis and taxonomy show in which scaled systems the Amdahl's law limits the speedup.

The rest of the paper is organised in several sections as follows. The speedup definitions and limits in parallel and distributed systems are described in Section II. Section III defines a taxonomy for scaled systems in cloud, in order to adapt the existing Amdahl's and Gustafon's laws for elastic services. According to the taxonomy, Section IV models the speeds and speedups for each scaled system for various load regions. Despite the virtualisation layer, the cloud environment can achieve even a superlinear speedup, as discussed in Section V. Section VI discusses further challenges. Finally, we conclude the paper in Section VII.

II. Background

Parallel and distributed systems offer a powerful environment that can be utilised for two main purposes: to speed up some algorithm's execution or to execute some big data problems. The former is useful in order to finish with execution in proper time; for example, we need today a weather forecast for tomorrow, and it is unusable to have it tomorrow. Distributed systems are used to solve a problem that cannot be even started on a single machine due to hardware limitation. Both parallel and distributed systems have more computing resources than a nominal single-machine or a single-processor system. In this paper, we will denote these systems as *scaled systems*.

Two main laws exist in the computer architecture, or more broader in the parallel and distributed systems, which limit the speedup that can be achieved, according the algorithm's type: Amdahl's and Gustafson's laws. Both laws target the speedup, but analyse it from different perspectives.

Let's analyze a scaled system with a scaling factor p. The metric for measuring the performance of a scaled computing system is the *speed* $V(p)$, which defines the amount of work $W(p)$ performed for a period of time $T(p)$, as presented in (1). Another important metric is the *normalised speed* $NV(p)$, which measures the amount of work per processor per time period, as defined in (2).

$$V(p) = \frac{W(p)}{T(p)} \tag{1}$$

$$NV(p) = \frac{V(p)}{p} = \frac{W(p)}{T(p) \cdot p} \tag{2}$$

To compare the scaled with a non-scaled system, one should evaluate the *speedup* $S(p)$, which is defined as a ratio of speeds of the scaled system and the best speed in the non-scaled system, as presented in (3).

$$S(p) = \frac{V(p)}{V(1)} = \frac{W(p)/T(p)}{W(1)/T(1)} \tag{3}$$

The amount of work is constant for fixed-time algorithms, which transfers (3) to (4), where $T(1)$ denotes the execution times of the best sequential algorithm, while $T(p)$ the execution time of the algorithm on scaled system with scaling factor p.

$$S(p) = \frac{T(1)}{T(p)} \tag{4}$$

Amdahl's Law limits the size of the problem and limits the speedup to the value $S_{max}(p) = 1/s$, where s is the serial part of the algorithm. As one can observe, the maximal theoretical value for the speedup is limited and does not depend on the number of processors. On the other side, Gustafson reevaluated the Amdahl's Law by showing that a *linear* speedup $S_{max}(p) = p$ can be achieved if a problem is executed within a fixed time. He achieved a near linear speedup of impressive 1000, when running a problem on 1024 cores [9].

III. A taxonomy of scaled systems

Usually both Gustafson's and Amdahl's laws are intended for granular algorithms, which can be divided into many independent sub-tasks and then scattered to a scaled system for execution. This section presents a taxonomy that we define for scaled systems in order to adapt both laws to be appropriate for cloud elastic services. We are using a similar approach for scalable algorithms, such as web services, with an exception that in this case, the parallelisation is usually not conducted by some API, but on the web server level.

III.1 Definition and classification of scaled systems

Let a *nominal system* be a cloud system that possesses R cloud resources and is loaded with L requests, as presented in Fig. 1 a). One would expect the nominal system can handle L amount of work in a specific time period using R resources. For example, the load can be represented as the number of requests for some service which is hosted in a group of VMs that have a total of R cloud computing resources.

Our classification is based on scaling both the requirements of a cloud computing system and cloud resources. Therefore, we define two scaling types in cloud computing: scaling the load (requirements) and scaling the (cloud computing) resources. A different scaling factor can be used for requirements and resources. Without loosing generality, we assume that the resources can scale $p > 1$ times, while the load, N times.

We will use the notation $xRyL$ to define the taxonomy of scaling the cloud systems where $x, y \in \{n, s\}$ are the indicators in front of each scaling parameter. The s presence indicates that the corresponding parameter is scaled and n if it is not scaled. According to this notation, the nominal system is defined as a *non-scaled Resources non-scaled Load*, and denoted as ($nRnL$) system.

If the customers want to improve the performance of a service hosted in a cloud system, they need to scale the cloud resources. In case of scaling the load, there are two possibilities for the customer: either to retain the cost (keep the same cloud resources), but degrade the performance, or to scale the cloud resources and to retain the same performance. Consequently, we will define three different scaled systems when only one or both parameters are scaled with Definitions 1, 2, and 3. All three scaled systems are presented in Fig. 1 b), c) and d).

Definition 1 ($sRnL$ scaled system) *The $sRnL$ scaled cloud system denotes a cloud system with* scaled Resources non-scaled Load, *that is, a system with p times more cloud resources.*

Definition 2 ($nRsL$ scaled system) *The $nRsL$ scaled cloud system denotes a cloud system with* non-scaled Resources scaled Load, *that is, a cloud system with N times more load.*

Definition 3 ($sRsL$ scaled system) *The $sRsL$ scaled cloud system denotes a system with* scaled Resources scaled Load, *that is, a system with p times more cloud resources and N times more load.*

The next examples explain these types of scaled systems. Assume that a web server hosted in a cloud instance with one CPU core ($R = 1$) can handle 100 client requests ($L = 100$) in acceptable response time. According to the Gustafson's Law one would expect that the performance would be doubled when the same 100 requests are executed on a server using resources with double the capacity ($sRnL$).

Another example is when both the load and resources are scaled, that is, the expected response time of 200 requests to be executed on a server with doubled resources should be the same as the nominal case - 100 requests executed on one CPU core ($sRsL$). And, for $nRsL$, the response time should be doubled if the load is increased to 200 requests.

III.2 Expected performance of scaled cloud systems

Let PF be a function (5) that returns the performance P of a system with specific resources R and loaded with a load L. Then, (6) defines the expected performance for all three scaled systems.

$$P = PF(R, L) \tag{5}$$

$$\begin{aligned} \textbf{sRnL}: & p \cdot P = PF(p \cdot R, L); \\ \textbf{nRsL}: & \frac{1}{p} \cdot P = PF(R, p \cdot L); \\ \textbf{sRsL}: & P = PF(p \cdot R, p \cdot L). \end{aligned} \tag{6}$$

This classification of scaling the system can help in determination of performance limits of a system.

IV. Theoretical analysis of scaled systems compliance with Amdahl's and Gustafson's laws

In order to adapt both laws for elastic services, this section introduces the work per resource and models the speedup for scaled systems compared to nominal (non-scaled) systems.

IV.1 Modeling the resource utilization

In order to find the *resource utilization* W we determine how much average work (load) L is sent to a particular resource R and calculate it according to (7) as a ratio of the load and the number of resources. This parameter shows the average "speed" of performing a particular work per resource. To simplify the notation, in the remaining text we will use abbreviations omitting the R and L identifications, such as nn for the $nRnL$ system.

$$W_{nn} = \left\lceil \frac{L}{R} \right\rceil; \tag{7}$$

Next, distribute all L requests in groups, such that each group has R requests to map each request to a specific computing resource. Then, in each time period, R requests will

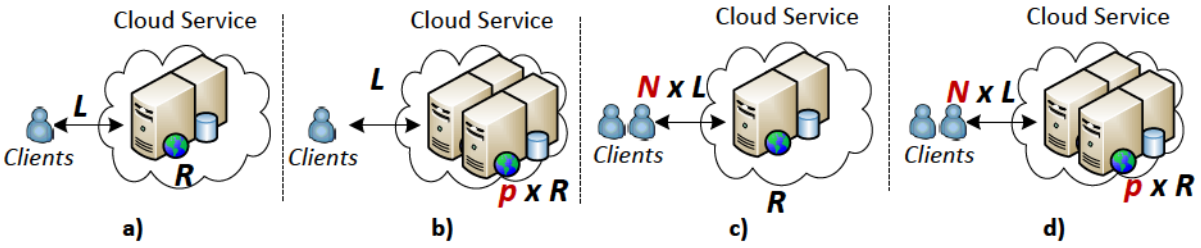

Figure 1: Nominal (non-scaled) system nRnL a) and three possible scalings b) sRnL, c) nRsL and d) sRsL

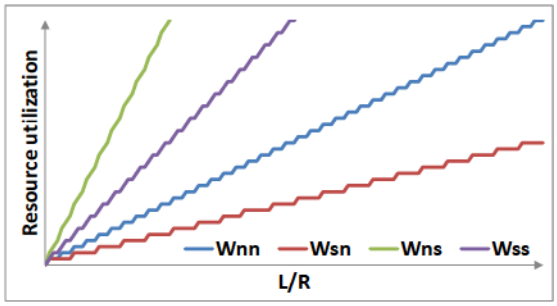

Figure 2: Resource utilization of the scaled systems

be scattered among available R resources, which yields to (7). Note that the last group will have $L \mod R$ requests, and not all the resources will be loaded with requests.

By applying the corresponding parameters for the three scaled systems into (7), then (8) defines the resource utilizations (expressed as work per resource) W_{sn}, W_{ns}, W_{ss}, corresponding to the $sRnL$, $nRsL$ and $sRsL$ systems.

$$W_{sn} = \left\lceil \frac{L}{p \cdot R} \right\rceil; \quad W_{ns} = \left\lceil \frac{N \cdot L}{R} \right\rceil; \quad W_{ss} = \left\lceil \frac{N \cdot L}{p \cdot R} \right\rceil; \tag{8}$$

Observing the definitions for all four work per resource, one can conclude that all of them depend on the ratio L/R, representing a nominal value of a work per resource. Therefore, we will continue with an analysis that determine the impact of the L/R ratio over the work per resource and the speedup.

Fig. 2 presents the resource utilization (work per resource) of the nominal and the three scaled systems. Observe that all resource utilizations are in a shape of stairs, with a linear trendline. The stairs' effect appears due to roundup function in (7) and (8). Obviously, the resources of $sRnL$ scaled system have a smaller amount of work, while the other two scaled systems have more, compared to the nominal system. Fig. 2 is obtained for a value of $p/N = 2/4 = 1/2$. We must note that W_{ss} and W_{ns} can change their place in Fig. 2, depending whether the p/N ratio is greater of smaller than 1.

IV.2 Modeling the speedup

In order to measure the impact of scaling, (9) defines the *speedup* S_{sn}, S_{ns}, S_{ss} for all scaled systems, when compared to the nominal one.

$$S_{sn} = \frac{\left\lceil \frac{L}{R} \right\rceil}{\left\lceil \frac{L}{p \cdot R} \right\rceil}; \quad S_{ns} = \frac{\left\lceil \frac{L}{R} \right\rceil}{\left\lceil \frac{N \cdot L}{R} \right\rceil}; \quad S_{ss} = \frac{\left\lceil \frac{L}{R} \right\rceil}{\left\lceil \frac{N \cdot L}{p \cdot R} \right\rceil}; \tag{9}$$

Fig. 3 visually presents all three speedups as a function of the L/R ratio, along with their trendlines. The speedup S_{sn} shows an increasing trend starting from 1, and saturates up to the scaling factor p when $\frac{L}{R} \to \infty$. Whenever p is a divisor of L/R, the speedup achieves its maximal value of p, regardless of the L/R ratio value, as depicted with point $A(i \cdot p, p)$. Although, seemingly, it looks like that the $sRnL$ scaled system relies on the Gustafson's Law, it is true only when the L/R ratio is huge. For smaller L/R ratio, scaling the resources will not provide a greater speedup, which is exactly the Amdahl's Law.

The speedup S_{ns} starts from $S_{ns} = 1$ and saturates its value to the point $1/N$ for greater ratio L/R. Obviously, although $S_{ns} < p$, and seemingly it is a sublinear speedup, in fact this is a slowdown. This is expected since the load is increased compared to the nominal system. According to Fig. 2, the work per resource is increased, which will reduce the performance. The tradeoff for this performance suffering is the constant cost that the customer should pay for renting the resources.This is the resource underprovisioning and overutilisation.

Similar behavior for speedup is present for the the $sRsL$ scaled system. The only difference is that the trendline saturates to the value $S_{ss} = p/N$. Let us discuss about the

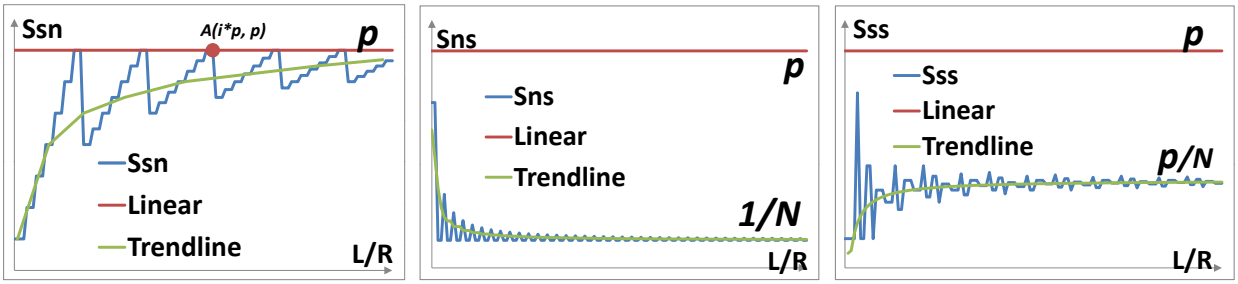

Figure 3: Speedup for various scaled systems as a function of L/R

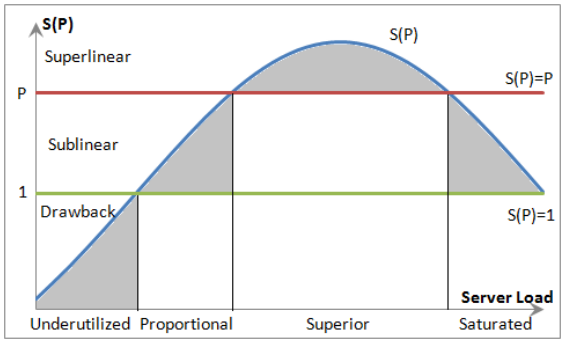

Figure 4: Expected speedup of a scaled system

expected performance of this scales system (6) and the calculated speedup (9). If the resource scaling follows the scaling of the work, then the speedup will saturate to 1, which means that this scaled system scales ideally. However, if $p > N$, which means overprovisioning and underutilisation, we are getting closer to the $sRnL$ scaled system, in the hands of Amdahl's Law.

IV.3 Going beyond the speedup limits

Although previous subsection presents the theoretical limits of the speedup in various scaled systems, several examples are reported where the speedup went beyond the limits, that is, a *superlinear* speedup is achieved.

Ristov et al. [10] have modeled the performance behavior of services classifying five sub-domains of speedup:

- *Drawback* $0 < S(p) < 1$ - worse performance for the scaled system;
- *No Speedup* $S(p) = 1$ - the new scaled system reports the same performance as unscaled;
- *Sublinear* $1 < S(p) < p$ - similar to Gustafson's scaled speedup;
- *Linear* $S(p) = p$ - maximum limited speedup according to the Gustafson's scaled speedup;
- *Superlinear* $S(p) > p$ - greater performance than the limited speedup.

The expected sub-domains, along with four regions of server load (underutilised, proportional, superior and saturated) are presented in Fig. 4 [10]. The first three regions are already expressed in our theoretical analysis. The superior region is of interest in this paper, and it appears because the web server with one core will enter in its saturation mode, while the scaled system is still in its normal mode. The superior region ends when the scaled system enters the saturation mode. This means that theoretical speedups of all scaled systems do not saturate to the constant value, but they will start to falling down when the ratio L/R will increase up to some level when even the scaled resources cannot handle the load in the appropriate time.

However, the reported results show that this model works only for both computation-intensive and memory-demanding web services, while the computation-intensive only web services achieve a sublinear speedup, that is, those systems have four regions.

V. Analysis of a superlinear speedup in cloud environment

Nowadays, cloud computing is being increasingly used for high-performance and high throughput applications. It allows the customers to rent, for example, 1000 processors and execute a certain task at peak times, instead of building their own data center. Since the cloud's pricing strategy is linear, and expected speedup is also linear, it seems that customers will be charged fairly. However, there are several

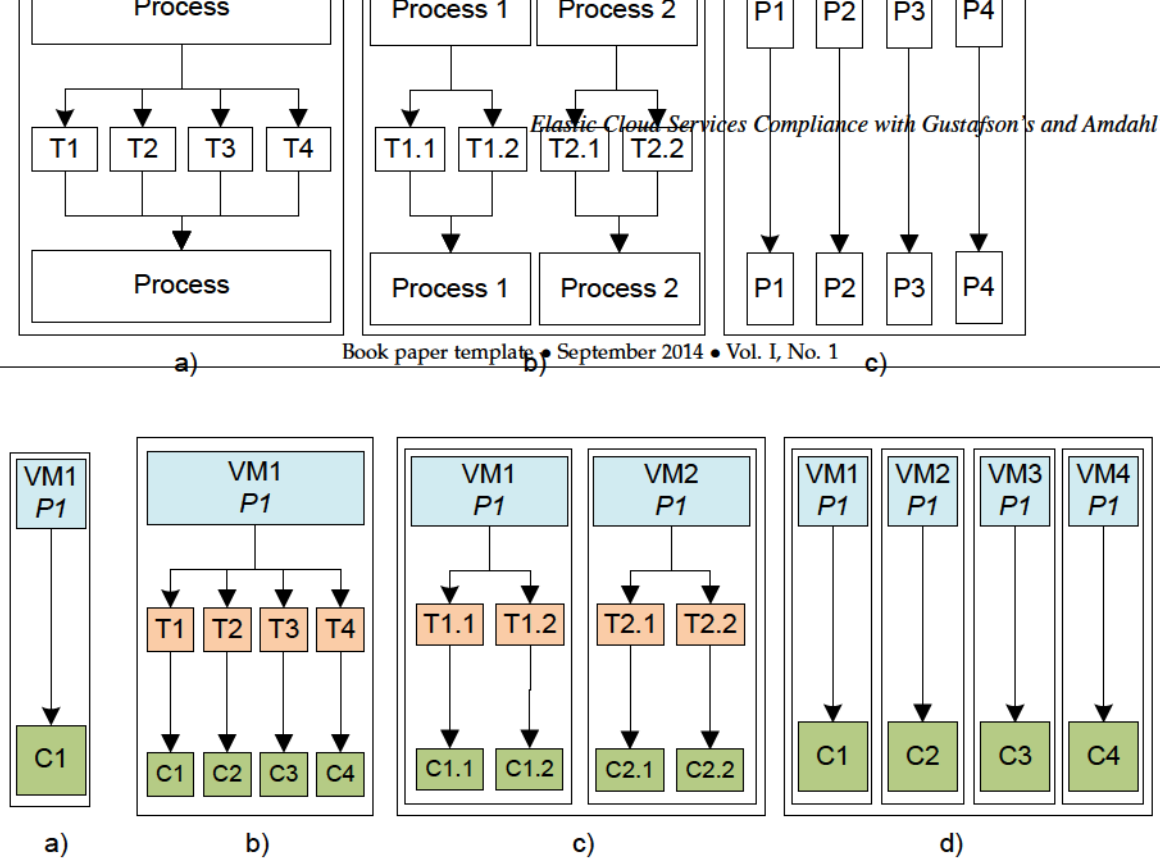

Figure 5: Example of b) Vertical, c) Diagonal, and d) Horizontal scaling of nominal resources a) for granular algorithms

cases where the superlinear speedup is achieved, despite the virtualisation layer.

V.1 Granular algorithms

Customers can scale their rented resources horizontally, vertically or diagonally in the cloud for the *sRnL* or *sRsL* scaled systems. If the original configuration maps one process to a VM instance hosted on a processor with one CPU core, as presented in Fig. 5 a), then Fig. 5 b), c) and d) present the three possible cloud scalings. The horizontal scaling presented in Fig. 5 d) increases the number of same VM instances and maps a separate process (with a single thread) to a different VM instance. The vertical scaling presented in Fig. 5 b) increases the number of CPU cores per VM (resized VM) and maps separate threads of a single process to a different core on the same VM instance. A combination of the both scaling types yields a diagonal scaling presented in Fig. 5 c). To realise the vertical and diagonal scaling, the customer should use some API for parallelization, such as OpenMP, which will create parallel threads.

Few papers are reporting a superlinear speedup in both the horizontal and vertical scaling. A superlinear speedup is reported for cache-intensive algorithms in [5] for the case of vertical scaling. Although sequential execution utilises cache in the sequential execution more, the superlinear speedup can be achieved also for horizontal scaling in the cloud, according [11]. The authors of [12] have determined that the cloud environment can handle the cases when the problem size can be fitted in the last level cache memory better leading to a superlinear speedup.

V.2 Scalable algorithms

Fig. 6 presents three possible ways how to scale from nominal system to the *sRnL* scaled system. Similarly, there is a horizontal, vertical and diagonal scaling. The main difference here is the necessity of cloud load balancer that will schedule the load among many end-point VM instances for horizontal and diagonal scaling.

Ristov et al. [6] proposed a scalable architecture for e-ordering system hosted in the cloud. Their experiments reported a significant superlinear speedup of 20 for the *SRNL* system with a scaling factor $p = 4$ analyzing the response time. The superlinearity also appeared for the throughput, i.e. the percentage of responses for given number of requests per second.

V.3 Superlinear speedup of a load balancer

Distributing the load to several end-point servers is much easier in the case of a load balancer, which will forward the load to the servers by using a particular algorithm. Since it is a new layer, it adds a small amount of delay, which usually depends on the load. However, Ristov et al. [13] developed a balancer with a region that achieves a superlinear speedup when using more end-point servers.

Even more, when it is used with only one end-point web server, the results are still better than the case without it. This appears because of the connections that are opened to the end-point web servers are maintained without opening a new connection for each client request. This reduces the number of resources for creating a new session compensating the delay produced due to the additional layer.

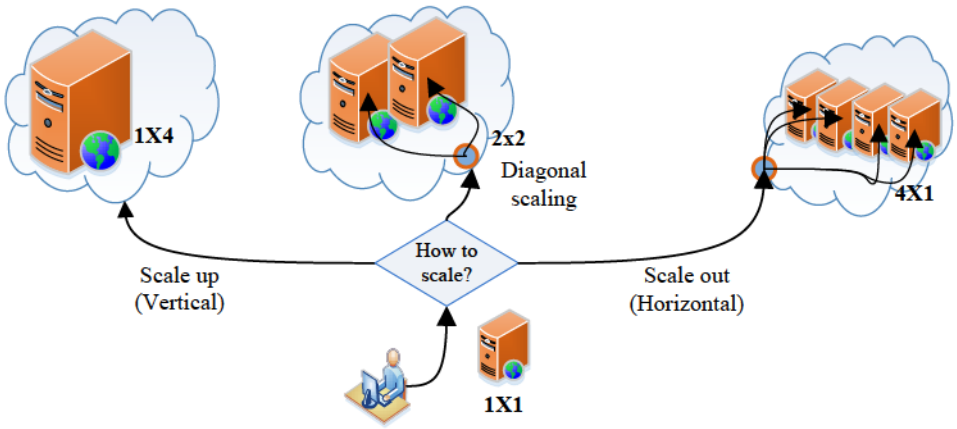

Figure 6: Example of vertical, diagonal and horizontal scaling of nominal system for elastic cloud services

VI. Discussion

This section discusses further challenges and issues connected to the performance of cloud elastic services.

VI.1 Granular and scaling algorithms' similarity

Although granular and scaling algorithms seem to be totally different, as they are executed in different environment, they still have several similarities. In case of persistent algorithms, more memory is needed for both algorithm types. That is why a sublinear speedup is achieved for computation-intensive web services [10].

We can observe another similarity. The granular algorithms executed on tightly coupled processors correspond to vertical scaling for scaling algorithms, while executed on loosely coupled processors to horizontal scaling.

These similarities can be used to utilise the pros of one algorithm type for the other, which could also lead to a superlinear speedup. In this context, we can follow the idea of Trang et al. [14] who introduced a model for parallel execution of web services promising that both approaches can be used together.

VI.2 New challenge: How to scale?

Cache-intensive granular algorithms, whose data reuse complexity is similar with the problem size, will benefit from a bigger cache. Many Intel's multiprocessors use a marketing trick based on a huge L3 smart cache. However, one can easily check that it is not shared among all cores, but only among part of them. For example, 6MB of total 12MB cache is shared between each group of two cores. In this case, the vertical scaling will utilise more the last level cache. AMD multiprocessors usually use a smaller L3 cache, but it is shared among all cores of the multiprocessor. Therefore, depending on the algorithm, appropriate processor and scaling type should be chosen in order to achieve the best speedup, potentially superlinear.

On the other hand, today's cloud elastic resources can also be scaled in different ways: horizontally, vertically or diagonally, each of which can offer various performance and possibility for achieving a superlinear speedup. The vertical scaling provides a better speedup, but the horizontal offers more flexible scaling of resources, which can minimise the cost. Although, using a load balancer in front of the siblings, a superlinear speedup can be achieved due to reduced number of opened connections.

VI.3 Further challenges

Achieving a superlinear speedup does not necessarily mean that customers will obtain the maximum achievement. In the workflow executions in parallel and distributed systems, customers usually use bi-objective optimizations to minimise the makespan and cost. These two parameters are opposite one to another. Minimising the makespan produces a greater cost and vice versa.

Cloud computing customers can set a deadline for the execution requiring a minimal cost, rather than a minimal makespan [15]. In these cases, budget constraints and reducing the race for the speedup can yield the reduced cost for the

execution. For example, although a superlinear speedup can be achieved in a Windows Azure cloud for matrix multiplication when VM instances with Windows operating system are used, Linux VM instances achieved better performance cost trade-off because they are cheaper.

On the other side, there is a risk of cloud resources performance variation, different setup time [16], instance failure over the time [17] and difficulty to predict the performance, which will harden the resource provisioning [18]. Additional problem in modeling the elastic cloud services' behavior is the uncertainty in cloud provisioning and VM instability. For example, Dejun et al. [19] reported a performance uncertainty of up to 8% in Amazon EC2.

Increasing the budget by duplicating the tasks on more than one instance could mitigate those risks, in order to meet the deadline [20]. Sometimes, using a bigger instance executes the task faster, rather than waiting several minutes for the deployment time to start another smaller, but an appropriate instance, which reduces the turnaround time of an activity [21].

Not all offered pricing models are linear. For example, some providers charge the customers on hourly based policy, while others charge some amount at the beginning plus charge then per smaller time unit. For example, Google charges the usage for the first 10 minutes, and then per minute. Also, Google have recently introduced the non-linear model by including the VM usage sustainability. All these issues impact on choosing the appropriate scaled system for a specific cloud elastic service.

VII. Conclusion

Cloud services are scalable and can be executed in both the parallel and distributed systems by load balancing among the scaled resources. This balancing reduces the amount of work per resource, which speedups the average execution time. Predicting and measuring the performance of such services is very difficult because the real cloud elastic service receives client requests with an unknown distribution probability function. Also, they are hosted on an unpredictable resource provisioning, which makes their modeling almost impossible. Still, by using the upper and lower limits of the speedup, one can compare the fairness of the pricing model.

The Amdahl's and Gustafson's laws set limits on the speedup that a scaled system achieves, but usually for granular algorithms. However, even in the traditional parallel and distributed systems, there are many cases when these laws are disproved due to the nature of the algorithms, hardware and software architecture. The uncertainty of the VM provisioning and performance, along with many differences between the scaling and granular algorithms, questions their compliance with both laws. However, our modeling and theoretical analysis showed that cloud elastic services are compliant with both laws. Such general laws are push drivers to enable the technologies and pull drivers that lead toward technical innovations. This chain of push and pull drivers makes the positive feedback that enables the overall technology continual development.

Acknowledgment

This work is partially supported by the European Union's Horizon 2020 research and innovation programme under the grant agreements 644179 ENTICE: dEcentralized repositories for traNsparent and efficienT vIrtual maChine opErations (first two authors) and 643946, CloudLightning: Self-organizing, self-managing Heterogeneous Clouds (fourth author).

The authors would like to acknowledge networking support by the COST programme Action IC1305, Network for Sustainable Ultrascale Computing (NESUS).

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[2] A. Gupta, D. Milojicic, Evaluation of hpc applications on cloud, in: Open Cirrus Summit (OCS), 2011 Sixth, 2011, pp. 22–26. doi:10.1109/OCS.2011.10.

[3] S. A. Tsaftaris, A scientist's guide to cloud computing, Computing in Science Engineering 16 (1) (2014) 70–76. doi:10.1109/MCSE.2014.12.

[4] L. Liu, M. Zhang, Y. Lin, L. Qin, A survey on workflow management and scheduling in cloud computing, in: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, 2014, pp. 837–846. doi:10.1109/CCGrid.2014.83.

[5] M. Gusev, S. Ristov, Superlinear speedup in Windows Azure cloud, in: Cloud Networking (IEEE CLOUDNET), 2012 IEEE 1st International Conference on, Paris, France, 2012, pp. 173–175.

[6] S. Ristov, F. Dimitrievski, M. Gusev, G. Armenski, Scalable system for e-orders as a service in cloud, in: International Conference on Computer as a Tool (IEEE EUROCON 2015), Salamanca, Spain, 2015, pp. 1–6.

[7] G. M. Amdahl, Validity of the single-processor approach to achieving large scale computing capabilities, in: AFIPS Conference Proceedings, Vol. 30, AFIPS Press, Reston. Va., Atlantic City, N.J., 1967, pp. 483–485.

[8] J. L. Gustafson, Reevaluating Amdahl's law, Communication of ACM 31 (5) (1988) 532–533.

[9] J. Gustafson, G. Montry, R. Benner, Development of parallel methods for a 1024-processor hypercube, SIAM Journal on Scientific and Statistical Computing 9 (4) (1988) 532–533.

[10] S. Ristov, M. Gusev, G. Velkoski, Modeling the speedup for scalable web services, in: A. M. Bogdanova, D. Gjorgjevikj (Eds.), ICT Innovations 2014, Vol. 311 of Advances in Intelligent Systems and Computing, Springer International Publishing, 2015, pp. 177–186.

[11] M. Gusev, S. Ristov, Resource scaling performance for cache intensive algorithms in Windows Azure, in: F. Zavoral, J. J. Jung, C. Badica (Eds.), Intelligent Distributed Computing VII, Vol. 511 of SCI, Springer International Publishing, 2014, pp. 77–86.

[12] M. Gusev, S. Ristov, The optimal resource allocation among virtual machines in cloud computing, in: Proceedings of The 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), Nice, France, 2012, pp. 36–42.

[13] S. Ristov, K. Cvetkov, M. Gusev, Implementation of a scalable L3B balancer, Scalable Computing: Practice and Experience 17 (2) (2016) 79–90. doi:10.1109/TE.2014.2327007.

[14] M. X. Trang, Y. Murakami, T. Ishida, Cloud Computing: 6th International Conference, CloudComp 2015, South Korea, Springer International Publishing, 2016, Ch. Modeling Parallel Execution Policies of Web Services, pp. 244–254.

[15] M. A. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE Transactions on Cloud Computing 2 (2) (2014) 222–235. doi:10.1109/TCC.2014.2314655.

[16] M. Mao, M. Humphrey, A performance study on the vm startup time in the cloud, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 423–430.

[17] F. Wu, Q. Wu, Y. Tan, Workflow scheduling in cloud: a survey, The Journal of Supercomputing 71 (9) (2015) 3373–3418. doi:10.1007/s11227-015-1438-4.

[18] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, E.-G. Talbi, Towards understanding uncertainty in cloud computing resource provisioning, Procedia Computer Science 51 (2015) 1772 – 1781.

[19] J. Dejun, G. Pierre, C.-H. Chi, Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops: International Workshops, ICSOC/ServiceWave 2009, Stockholm, Sweden, November 23-27, 2009, Revised Selected Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, Ch. EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications, pp. 197–207.

[20] R. N. Calheiros, R. Buyya, Meeting deadlines of scientific workflows in public clouds with tasks replication, IEEE Transactions on Parallel and Distributed Systems 25 (7) (2014) 1787–1796. doi:10.1109/TPDS.2013.238.

[21] M. Mao, M. Humphrey, Scaling and scheduling to maximize application performance within budget constraints in cloud workflows, in: Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, 2013, pp. 67–78. doi:10.1109/IPDPS.2013.61.